

Dynamic Frontend Components

[Nike E-Commerce]

Introduction:

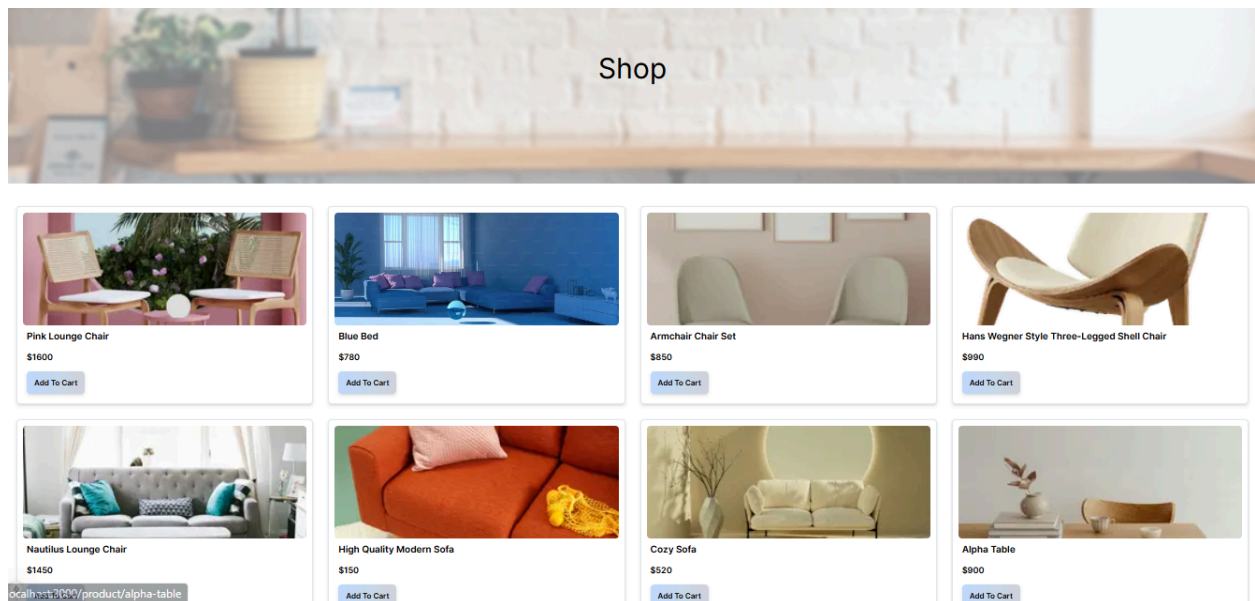
This niche e-commerce web project is built using Next.js and styled with Tailwind CSS, ensuring a responsive UI. It features navigation, product displays, and a smooth checkout experience.

Steps for Implementation:

Steps Taken:

1. **Project Setup** – Initialized Next.js app, configured Tailwind CSS, and set up Sanity for CMS.
2. **Component Development** – Built reusable UI components for product listings, cart, and checkout.
3. **API Integration** – Connected fake APIs to fetch product details, reviews, and user data.
4. **State Management** – Used React Context for global state handling (cart, user authentication).
5. **Deployment** – Optimized for performance and deployed using Vercel for scalability.

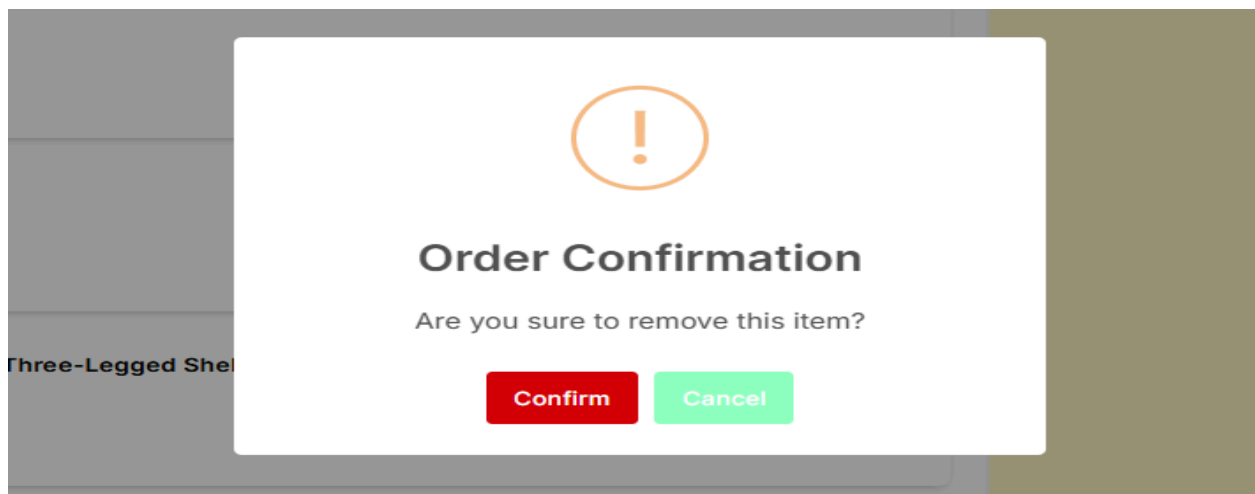
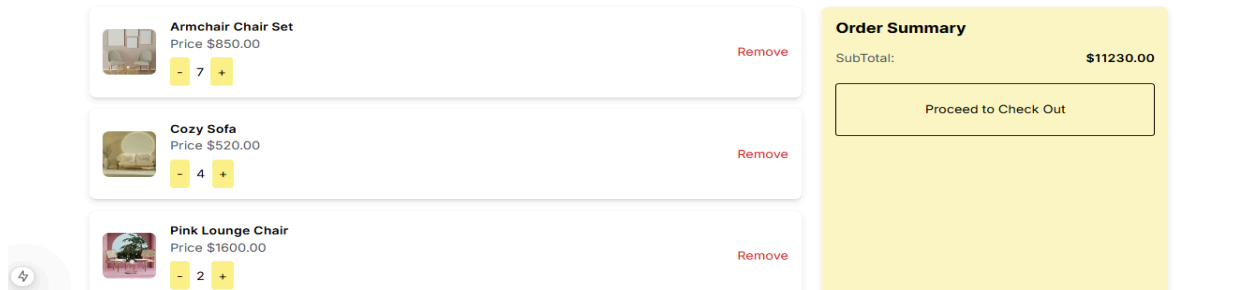
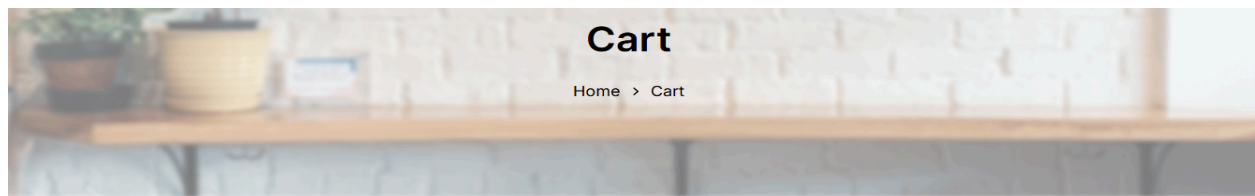
1-Product Listing Page

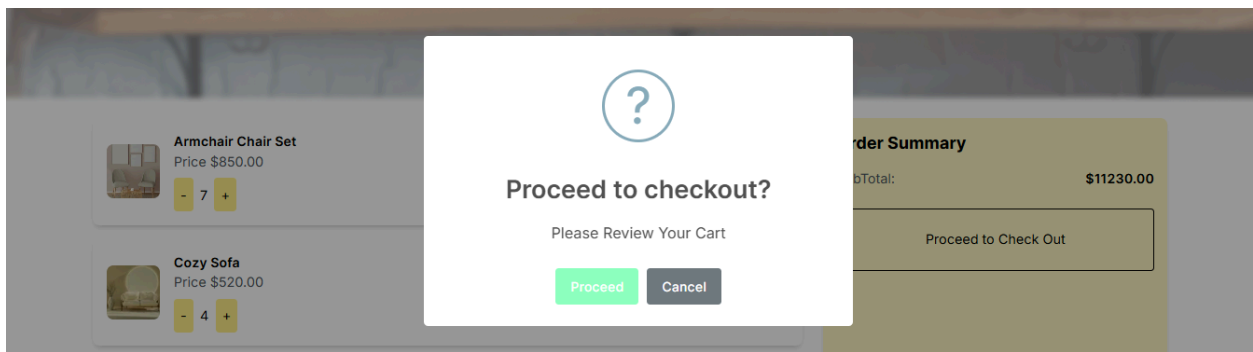


```
export default function Shop() {
  <div className="grid grid-cols-1 md:grid-cols-4 gap-8">
    {product.length > 0 ? (
      product.map((product) => (
        <div key={product.id}
          className="border rounded-lg shadow-md p-3 hover:shadow transition duration-200">
          <Link href={`/product/${product.slug.current}`}>
            <div className="">
              {product.image && (
                <Image
                  src={urlFor(product.image).url()}
                  alt="image"
                  width={200}
                  height={200}
                  className="w-full h-60 object-cover rounded-lg" />
              )}
            </div>
            <h1 className="font-semibold m-2" style={{ fontSize: "20px" }}>
              {product.name}</h1>
            <div className="text-lg font-semibold p-2">
              {product.price ? `$$${product.price}` : "price not available"}</div>
            <div className="p-2">
              <button className="bg-gradient-to-r from-blue-200 to-gray-300 text-black fo
                py-3 px-4 rounded-lg shadow-md hover:shadow-lg hover:text-gray-600 hover:scale-
                transition-transform duration-300 ease-in-out"
                onClick={() => handleCartButton(e, product)}>

```

2-Cart





```
3
4
5 export const addToCart = (product : Product) => {
6   const cart : Product[] = JSON.parse(localStorage.getItem('cart') || '[]')
7   const existingItemIndex = cart.findIndex(item => item._id === product._id)
8
9   if(existingItemIndex > -1){
10     cart[existingItemIndex].stocklevel += 1
11   }
12   else{
13     cart.push({
14       ...product, stocklevel: 1
15     })
16   }
17
18   localStorage.setItem('cart', JSON.stringify(cart))
19 }
20
21 export const removeItem = (itemId : string) => {
22   let cart : Product[] = JSON.parse(localStorage.getItem('cart') || '[]')
23   cart = cart.filter(item => item._id !== itemId)
24
25   localStorage.setItem('cart', JSON.stringify(cart))
26 }
27
28 export const updateQty = (itemId :string, quantity:number) => {
29   const cart : Product[] = JSON.parse(localStorage.getItem('cart') || '[]')
30   const itemIndex = cart.findIndex(item => item._id === itemId)
```

```
13 const CartPage = () => {
14   </div>
15
16   <div className="container w-full mt-40">
17     {cartItems.length === 0 ?(
18       <p className="text-center text-blue-600"> Empty Cart!</p>
19     ): (
20       <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
21         <div className="lg:col-span-2">
22           {cartItems.map((item) => (
23             <div key={item._id}
24               className="flex text-black items-center shadow-md rounded-lg p-4 mb-4">
25               <Image
26                 src={urlFor(item.image).url()}
27                 className="w-16 h-16 object-cover rounded-lg"
28                 alt= "Image"
29                 width={80}
30                 height={580}/>
31             <div className="flex-1 ml-4">
32               <h2 className="text-1 font-semibold">{item.name}</h2>
33               <p className="text-gray-600">Price ${item.price.toFixed(2)}</p>
34               <div className="flex items-center space-x-2 mt-2">
35                 <button
36                   onClick={() => handleQtyDecrease(item._id)}
37                 />
38               </div>
39             </div>
40           )>
41         </div>
42       </div>
43     )}
44   </div>
45 }
```

3-Related Products

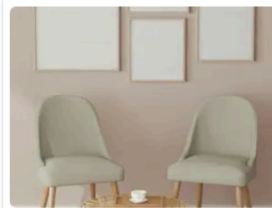
Related Products



Pink Lounge Chair
\$1600



Blue Bed
\$780



Armchair Chair Set
\$850



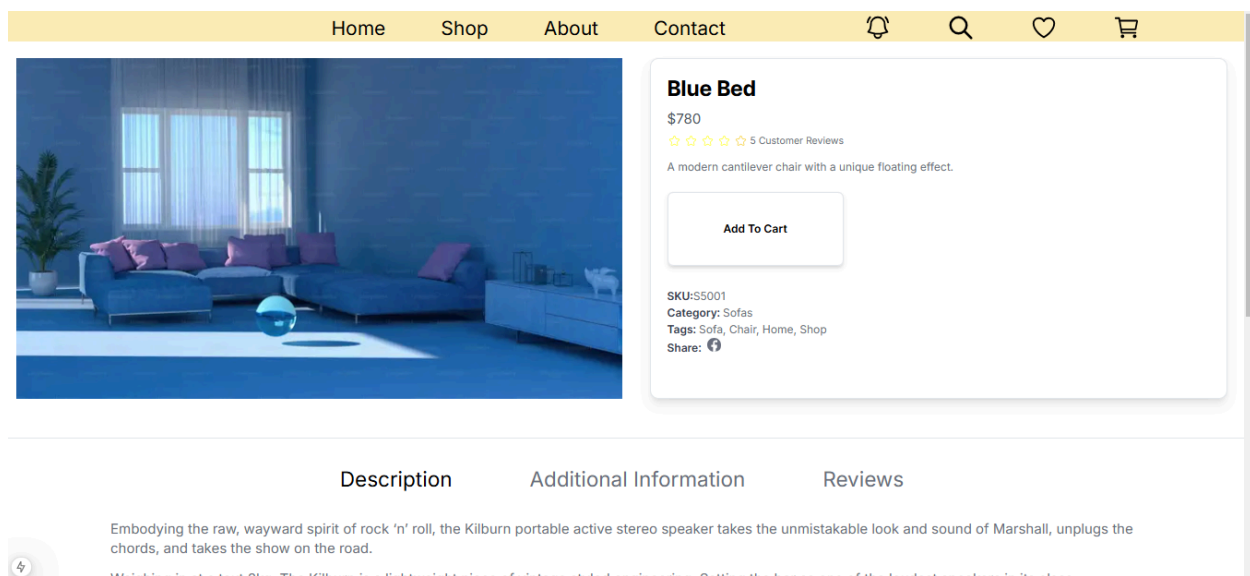
Hans Wegner Style Three-Legged Shell Chair
\$990

4

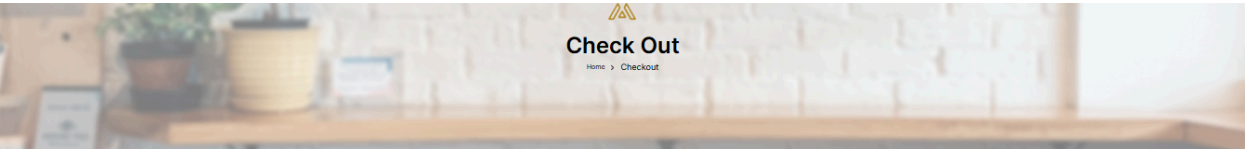
View More


```
1 use client
2 import { client } from "@sanity/lib/client";
3 import { urlFor } from "@sanity/lib/image";
4 import { allproducts, four } from "@sanity/lib/queries";
5 import { Product } from "@types/products";
6 import Image from "next/image";
7 import React, { useEffect, useState } from "react";
8
9 Debug this code | Explain this function to me | Optimize this function's Speed | Generate Doc String | Generate Unit test
10 const Furniture = () => {
11   const [product, setproduct] = useState<Product[]>([])
12
13   useEffect(() => {
14     Debug this code | Explain this function to me | Optimize this function's Speed | Generate Doc String | Generate Unit test
15     async function getproducts() {
16       const fetchedProduct: Product[] = await client.fetch(four)
17       setproduct(fetchedProduct)
18     }
19     getproducts()
20   }, [])
21
22   return (
23     <div className=" grid grid-cols-1 md:grid-cols-4 m-6 justify-center gap-8">
24       {product.map((product) => (
25         <div key={product._id}
26           className="border shadow-md p-2 hover:shadow transition h-40 duration-200"
27           style={{ height: "400px" }}>
28           {product.image && (
```

4-Individual Product Page



5-Checkout Page





Check Out

[Home](#) > [Checkout](#)

Billing Details

First Name

Last Name

Company Name (optional)

Country / Region

Street Address

Town / City

Province

Select a province

Zip Code

Phone

Order Summary

Armchair Chair Set	\$ 5950
Cozy Sofa	\$ 2080
Pink Lounge Chair	\$ 3200
Subtotal:	\$ 11230
Total:	\$ 11230

Your personal data will be used to process your order, manage access to your account

☒ Direct Bank Transfer

☐ Cash On Delivery

Your personal data will be used to support your experience throughout this website, to manage access to your account, and for other purposes described in our [privacy policy](#).

Place Order

DAY-4-HACKATHON

> .next

> app

> Cart

page.tsx

checkout

page.tsx

components

Shop

page.tsx

Ui

addtocartbtn.tsx

AddToCartButton.tsx

footer.tsx

navbar.tsx

paymentOption.tsx

prac-mainpg-items.tsx

sofa-practice.tsx

fonts

product\[slug]

page.tsx

studio

TS cartfunc.ts

favicon.ico

globals.css

```
1  "use client";
2  import { Product } from "@types/products";
3  import React, { useEffect, useState } from "react";
4  import { getCartItem, } from "../cartfunc";
5  import Navbar from "../components/navbar";
6  import { GrFormNext } from "react-icons/gr";
7  import Footer from "../components/footer";
8  import Link from "next/link";
9  import PaymentOptions from "../components/paymentOption";
10
11
12  const CheckOut = () => {
13    const [cartItems, setCartItems] = useState<Product[]>([])
14    const [discout, setDiscount] = useState<number>(0)
15    const [formValues, setFormValues] = useState({
16      FirstName: "",
17      LastName: "",
18      CompanyName: "",
19      CountryRegion: "",
20      StreetAddress: "",
21      TownCity: "",
22      Province: "",
23      ZipCode: "",
24      Phone: "",
25      EmailAddress: "",
26
27    });
28    const [formErrors, setFormErrors] = useState({
```

Errors:

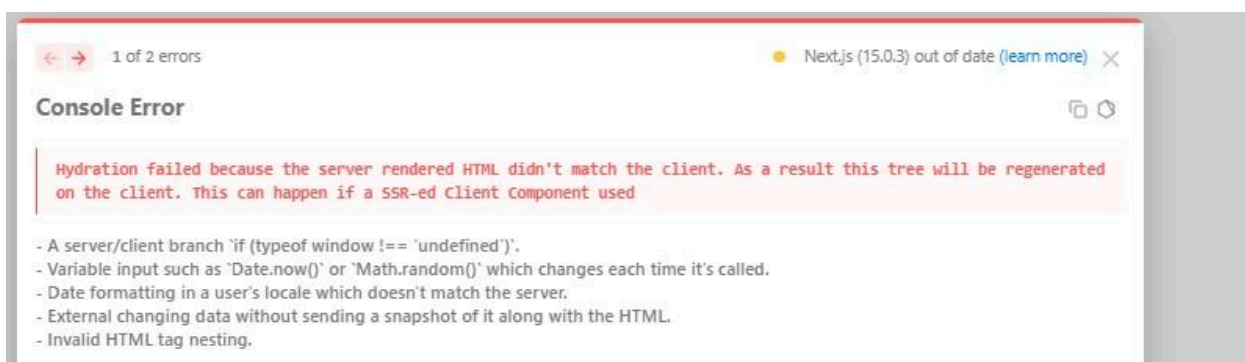
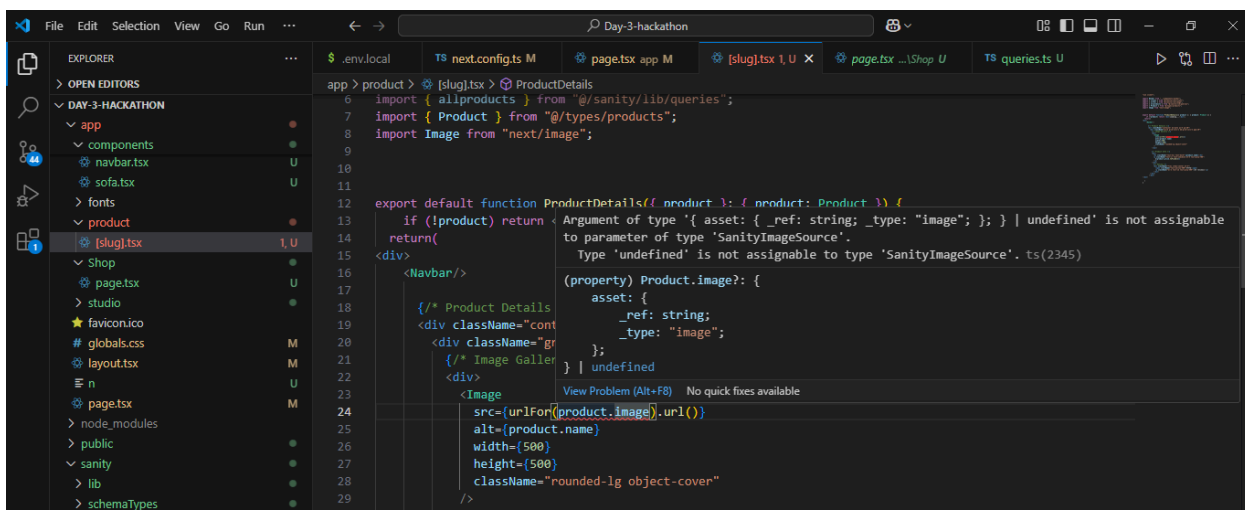
```
Error:  x 'const' declarations must be ini
[F:\GIAIC-hackaton\Day-3-hackathon -
55
56   }
57   // {quantity calculated}
58   const TotalQty () => {
59
60       return(
61           cart
62       )
63
64   }
65
66   return(
67       <div>
68         <div>
69           [sanityImageSource]
70         </div>
71       </div>
72     )
73   }
74 }
75
76 export default ProductDetails
```

x Expected a semicolon

```
[F:\GIAIC-hackaton\Day-3-hackathon -
55
56   }
57   // {quantity calculated}
58   const TotalQty () => {
59
60       return(
61           cart
62       )
63
64   }
65
66   return(
67       <div>
68         <div>
69           [sanityImageSource]
70         </div>
71       </div>
72     )
73   }
74 }
75
76 export default ProductDetails
```

x Return statement is not allowed here

```
[F:\GIAIC-hackaton\Day-3-hackathon -
63
64   }
65
66   return(
67       <div>
68         <div>
69           [sanityImageSource]
70         </div>
71       </div>
72     )
73   }
74 }
75
76 export default ProductDetails
```



Challenges Faced & Solutions Implemented

As a beginner in web development, I encountered several challenges while working on this project. One major issue was correctly using **Client Components** in Next.js. Initially, hooks like `useEffect` and `useState` threw errors because the "use client" directive wasn't properly placed. After some debugging, I ensured that it was the **first line** in the necessary files.

1- Hydration Mismatch Issue

- **Challenge:** The UI rendered differently between the server and client, causing hydration errors.
- **Solution:** Ensured all dynamic data fetching happened inside `useEffect` to prevent SSR conflicts.

2- Invalid next/image Hostname

- **Challenge:** Images from Sanity's CDN weren't loading due to missing configuration.
- **Solution:** Updated `next.config.js` to allow `cdn.sanity.io` under `images.domains`.

3- Dynamic Routing for Product Pages

- **Challenge:** The Card component wasn't receiving the correct data because it wasn't directly connected to Home or Shop pages.
- **Solution:** Used `StaticParams` (if using App Router) to ensure correct pre-rendering of dynamic product pages.

Best Practices Followed

1. **Backups** – At the end of each task, I made a copy of the project folder. This ensured that if anything broke, I always had a working version to fall back on.
2. **Following Next.js Guidelines** – I carefully separated **Server Components** and **Client Components**.
3. **Reading Error Messages Carefully** – Instead of guessing, I **read error messages line by line** to understand the issue and apply targeted fixes. This helped me debug efficiently and learn from mistakes.